Implementing Connected Component Labeling as a User Defined Operator for SciDB

Amidu Oloso[1,2], Kwo-Sen Kuo[1,3,4], Thomas Clune[1]
[1]NASA GSFC, Greenbelt, MD, USA,
[2]SSAI, Greenbelt MD, USA,
[3]Bayesics, LLC, Bowie, MD, USA
[4]University of Maryland, College Park, MD, USA
{amidu.o.oloso, kwo-sen.kuo, thomas.l.clune}@nasa.gov

Paul Brown, Alex Poliakov
Paradigm4
Waltham, MA, USA
{pbrown, apoliakov}@paradigm4.com

Hongfeng Yu
University of Nebraska - Lincoln
hfyu@unl.edu

***Abstract*—We have implemented a flexible User Defined Operator (UDO) for labeling connected components of a binary mask expressed as an array in SciDB, a parallel distributed database management system based on the array data model. This UDO is able to process very large multidimensional arrays by exploiting SciDB's memory management mechanism that efficiently manipulates arrays whose memory requirements far exceed available physical memory. The UDO takes as primary inputs a binary mask array and a binary stencil array that specifies the connectivity of a given cell to its neighbors. The UDO returns an array of the same shape as the input mask array with each foreground cell containing the label of the component it belongs to. By default, dimensions are treated as non-periodic, but the UDO also accepts optional input parameters to specify periodicity in any of the array dimensions. The UDO requires four stages to completely label connected components. In the first stage, labels are computed for each subarray or chunk of the mask array in parallel across SciDB instances using the weighted quick union (WQU) with half-path compression algorithm. In the second stage, labels around chunk boundaries from the first stage are stored in a temporary SciDB array that is then replicated across all SciDB instances. Equivalences are resolved by again applying the WQU algorithm to these boundary labels. In the third stage, relabeling is done for each chunk using the resolved equivalences. In the fourth stage, the resolved labels, which so far are "flattened" coordinates of the original binary mask array, are renamed with sequential integers for legibility. The UDO is demonstrated on a 3-D mask of $O(10^{11})$ elements, with $O(10^{8})$ foreground cells and $O(10^{6})$ connected components. The operator completes in 19 minutes using 84 SciDB instances.**

Keywords-Connected Component Labeling; User Defined Operator; SciDB; MemArray; Weighted Quick Union; array; mask; connectivity; equivalencies

I. INTRODUCTION

SciDB [1] is an open-source all-in-one data management and advanced analytics platform that features complex analytics inside a next-generation parallel array database. It is based on shared-nothing architecture for data parallelism, data versioning and provenance. Because it is array-based, SciDB is more suitable for scientific data analytics than traditional Relational Database Management Systems (RDBMS's). It provides extensive and flexible operators that can be efficiently "wired" together for more complex operations. SciDB is also extensible through User Defined Types (UDTs), User Defined Functions (UDFs) and User Defined Operators (UDOs).

This paper focuses on developing a UDO that performs Connected Component Labeling (CCL) of a binary mask stored as a SciDB array. Typically, such a mask array will be the result of performing some thresholding on other SciDB array(s) using a combination of operators/functions. For our specific examples, we are interested in studying the climatology of extreme weather events, e.g. winter blizzards, that may be present in multi-decadal data generated by climate models or retrieved from ground- and space-based instruments. In order to study the climatology of these events, the first step is to locate them and then track the evolution of each of them in space and time. Each of these events are represented by a group of connected cells spanning space and time. CCL is used to uniquely label each group of connected cells as a separate event. This work is conducted under the auspices of a larger project where we are using SciDB to integrate data storage and analysis. This paper also documents the presentation in [2].

II. IMPLEMENTATION DETAILS

A. Requirements

The use of CCL is ubiquitous in image processing where images are scanned and pixels are grouped into components based on some heuristic pixel connectivity. However, whereby images are usually represented by two-dimensional binary masks (or pixels), providing SciDB with CCL capability logically requires an implementation that will support arbitrary number of dimensions. It is thus important for the resulting UDO to have usability comparable with other SciDB operators where arrays of arbitrary dimensions are easily handled. Moreover, since our interest lies primarily in earth science applications where we may have periodic boundary conditions and/or may desire to enforce connectivity in any combination of the dimensions, our implementation must provide means to support these. Finally, our implementation must leverage SciDB's efficient ability to

handle larger-than-physical-memory arrays. This is particularly important for the CCL computation given the transitive nature of connectivity that induces large memory footprint.

B. Steps for meeting the requirements

- *Arbitrary dimensionality*: Each foreground pixel is uniquely identified by the pixel's flattened coordinate. This makes it easier to generically create pairwise neighbors that satisfy some specified connectivity, including periodic boundary cells that are adjacent. Connected pairwise neighbors is key input to the Weighted Quick Union (WQU) with half-path compression algorithm [3] used for the CCL computation. For an N-dimensional array, this identifier is computed as

$$P_{index} = x_{N-1} + \sum_{d}^{N-2}(x_d \prod_{r=d+1}^{N-2} D_r) \qquad (1)$$

  where P_{index} is the flattened coordinate for cell at coordinates $(x_i, i = 0, 1, \ldots N-1)$. D_r is the extent of dimension along coordinate r.
- *Flexible connectivity*: This is achieved by an auxiliary array of the same dimension as the mask array but with an extent of three (3) in each dimension. For example, for 2-D, the connectivity array can be plotted as shown in Fig. 1.
- *Larger than physical memory arrays*: SciDB provides *MemArray*, a data object with associated methods for handling large arrays distributed in chunks across all instances. It allows for each chunk to be operated on at a time by each instance. Parallelism comes from all instances working concurrently. This mechanism is used extensively in our implementation to manage memory usage.

C. Implementation Description

The implementation is illustrated by the simple example 2-D mask array shown in Fig. 2 where foreground cells are in black. The complete array is distributed into four chunks, each occupying a quadrant in Fig. 2. In general, chunk assignment to instances is managed by SciDB using a heuristic hash function. 4-connectivity is assumed for this illustration. Each foreground cell is first uniquely represented

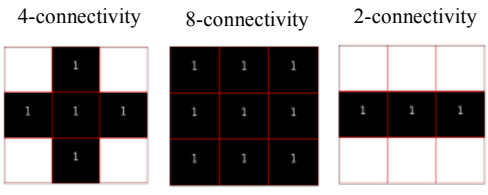

Figure 1. Sample 2D Connectivity Arrays.

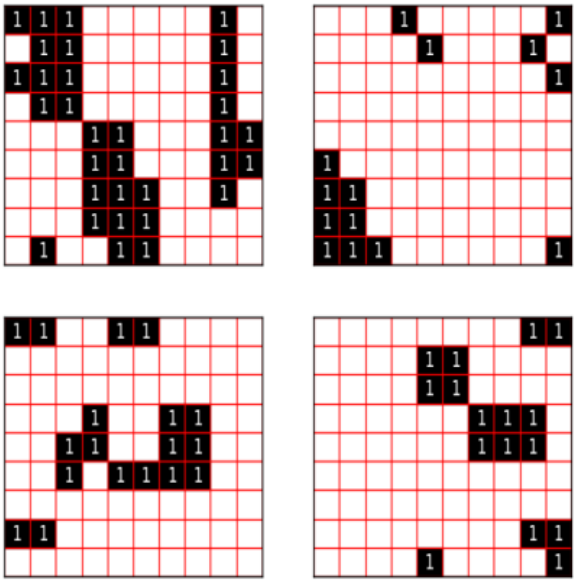

Figure 2. 2-D Mask Array for demonstrating the UDO CCL implementation

by its flattened coordinate as described in Section II(B).

There are four stages to the implementation. In Stage 1, each instance computes labels for its chunks, one chunk at a time. Parallelism is achieved by all instances proceeding concurrently. In this stage, the 4-connectivity auxiliary array is applied to each foreground mask to determine pairs of adjacent cells. Duplicate pairs are eliminated. The WQU algorithm is applied to each pair to determine the CCLs for the chunk. The CCLs are then written to *MemArray*. Fig. 3 shows the CCLs for each chunk after the first stage.

In Stage 2, label equivalencies are resolved. The *MemArray* from Stage 1 is parsed for boundary labels. These boundary labels are written into a new *MemArray* that is then replicated on all instances such that each instance has a global view of all boundary labels. Like in the first Stage, the 4-connectivity auxiliary array is applied to each label to produce pairs of connected boundary labels and the WQU algorithm is applied to generate CCLs of labels that are connected along the chunk boundaries. These resolved boundary labels are written to a 1-D *MemArray* indexed by boundary labels from Stage 1 to ensure fast access during the next stage i.e. Stage 3. A 2-D rendering of the resolved boundary labels is shown in Fig. 4.

In Stage 3, the labels from Stage 1 are relabeled and written to a new MemArray by propagating the resolved boundary labels inward as necessary using the 1-D *MemArray* from Stage 2 as follows:

```
if stage2_array[stage1_label] ! NULL :
        stage3_label <- stage2_array[stage1_label]
else :
        stage3_label <- stage1_label
```

Fig. 5 shows the result after Stage 3.

For the fourth and final stage, a relabeling is done such

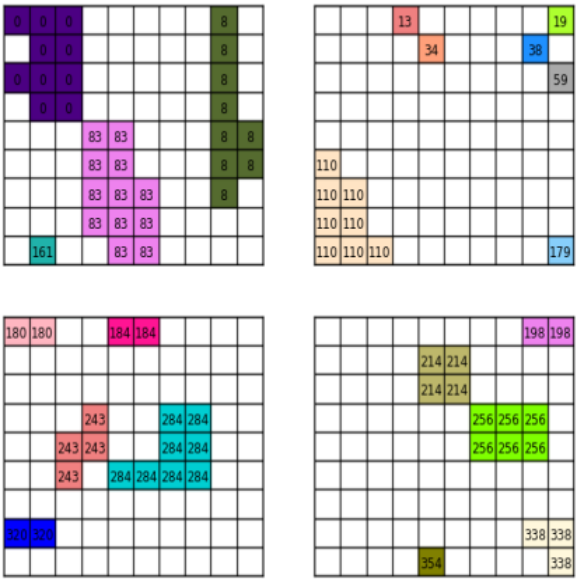

Figure 3. CCLs after Stage 1

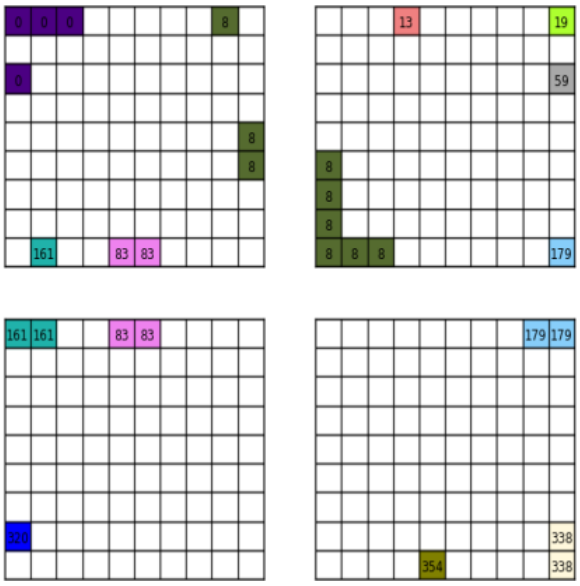

Figure 4. Boundary CCLs after Stage 2

that the out-of-order labels (represented by a subset of the "flattened" coordinates of the original mask array) are now represented by sequential integers for legibility. The result of this stage is shown in Fig. 6.

The new CCL UDO, named "ccl" is invoked as follows via SciDB's Array Functional Interface (AFL) as:

```
iquery -anq "store(ccl(mask_array, con_array), ccl_array)"
```

where "ccl" is the new CCL UDO, "store" is the SciDB native operator to write results into an array, "mask_array" is the binary mask array, "con_array" is the connectivity array and "ccl_array" is the array into which the resulting CCLs are written.

If boundary conditions are periodic, say in both directions, the new operator will be invoked with additional (integer) parameters such as:

```
iquery -anq "store(ccl(mask_array, con_array, 1, 2), ccl_array)"
```

where the additional parameters "1" and "2" simply means periodicity should be applied in both directions. For our 2-D example, this invocation produces the labels shown in Fig. 7.

III. Real Life Use Case

The motivation for our real life application was to study the climatology of winter blizzards present in the gridded GEOS-5 MERRA hourly data spanning a 37-year period from January 1, 1979 to December 31, 2015, where GEOS-5 stands for the Goddard Earth Observing System Model, Version 5 [4], and MERRA stands for Modern-Era Retrospective analysis for Research and Applications [5]. MERRA has a horizontal spatial resolution of ⅔° in longitude and ½° in latitude. Thus, this amounts to approximately ~24×365×37=324,120 time slices, each containing 540×361=194,940 grid cells, totaling ~6.32×10^{10} grid cells. Each cell, however, contains multiple variables that need to be processed in our use case.

The details for computing the mask array where foreground cells are identified as those meeting a blizzard threshold have been presented in [6]. The baseline SciDB array used to compute the mask array in [6] was created by assembling necessary variables (or attributes in SciDB terminology) pulled from four hourly data sets of GEOS-5 MERRA namely: MAT1NXFLX [7], MAT1NXLND [8], MAT1NXSLV [9] and MAC0NXASM [10]. The focus of the present work was to study the viability of the CCL UDO for identifying blizzards as distinct events. An event in this context is a group of blizzard cells connected in spatial and temporal coordinates by a defined connectivity criterion represented by the auxiliary connectivity array.

Consequently the mask array generated by [6] is 3-D i.e. 2-D space and 1-D time and it is $O(10^{11})$ in size. The number of foreground cells is $O(10^{8})$. The connectivity array used is the 3-D equivalent of the 4-connectivity array of the 2-D case shown in Fig. 1, i.e. two cells are considered connected if they share a same facet. The new CCL UDO is able to accurately compute all labels totaling over one million, i.e. $O(10^{6})$. Fig. 8 shows the presence of two blizzard events in central and eastern United States at hours 1330Z and 1430Z on February 5, 2010. Fig. 9 shows the same events at 18:30Z and 19:30Z with the event to the west already dissipated at 19:30Z. The top rows of Fig. 8 and Fig. 9 show the mask array while the bottom rows show the events accurately identified by the CCL UDO.

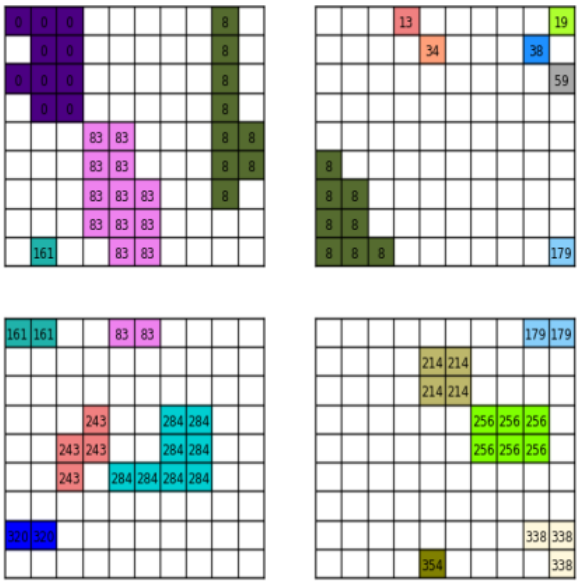

Figure 5. CCLs after Stage 3

Figure 6. CCLs after Stage 4 (Final Stage)

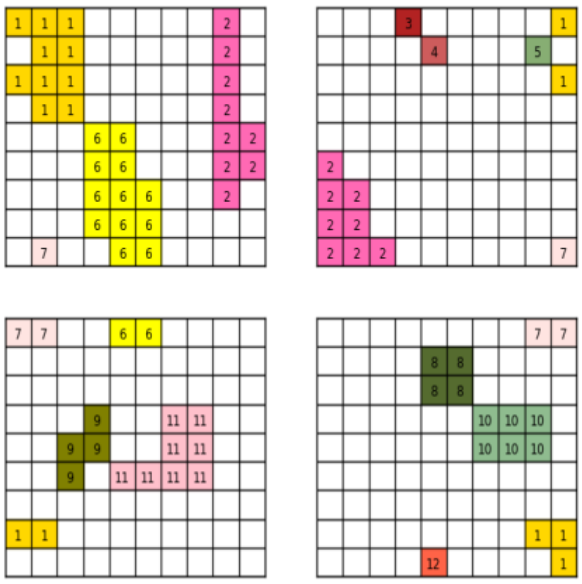

Figure 7. CCLs with periodic boundary conditions

IV. Computational Performance

Table 1 shows the wall times, as a range, for the different stages of the CCL UDO for the real life use case. The lower number is for the fastest instance and the higher number for the slowest. It should be noted that the reported times include, as necessary, computation of connected pairs, computation of the CCL, reading from *MemArrays* and writing to *MemArrays*. As expected, with the exception of stage 2, the scalability is reasonable. The equivalency resolution part of stage 2 that does not scale also happens to dominate the computational cost. This is because the foreground boundary cells for all chunks are involved in this part of the work. We are looking for ways to improve the scalability for this phase to reduce the total wall time. One critical goal that we have achieved is that the operator can compute CCL for arbitrarily large mask arrays within a reasonable amount of time without running out of physical memory, which our previous implementations failed to accomplish. For our use case, it took under twenty minutes to compute the CCL using 84 SciDB instances.

V. Concluding Remarks

We have implemented a UDO to compute CCL within SciDB. The UDO has been demonstrated on a real life use case where over a million labels were computed in under twenty minutes to represent blizzard events in 37 years of hourly GEOS-5 MERRA data. The UDO is able to leverage SciDB's *MemArray* to handle larger than physical memory cases. While most part of the UDO scales with number of instances, the equivalency resolution i.e. stage 2 which dominates the wall time requires significant improvement. This is a subject for future effort.

Additionally, the arrays that we deal with continue to grow in the time dimension. It is impractical and unreasonable to re-compute all labels each time we add new time slices especially since most events will last only a limited number of time steps. Therefore, it is important to only compute labels that have not been completed in the previous computations. Hence, another area of future effort is to augment the CCL UDO with the ability to exclude labels that have been completely resolved previously from any later computations due to the addition of new time slices.

TABLE I. Wall Times for the Stages of the CCL UDO (seconds)

Stage	Number of Instances			
	14	**28**	**56**	**84**
1	88 - 114	41 – 54	21 - 27	12 – 18
2 (replicate)	9 – 12	7 – 9	6 - 12	6 – 14
2 (resolve)	725 - 940	715 - 950	705 - 935	705 – 975
3	240 - 320	115 - 160	85 - 120	33 – 55
4	364 - 475	180 - 240	90 - 125	49 – 81

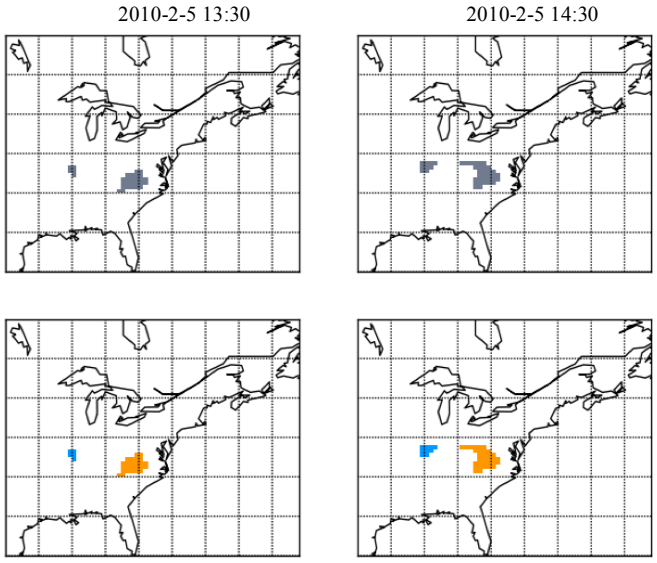

Figure 8. Two blizzard events at 1330Z and 1430Z of February 5, 2010. Top row: binary mask. Bottom row: the events identified by the new UDO

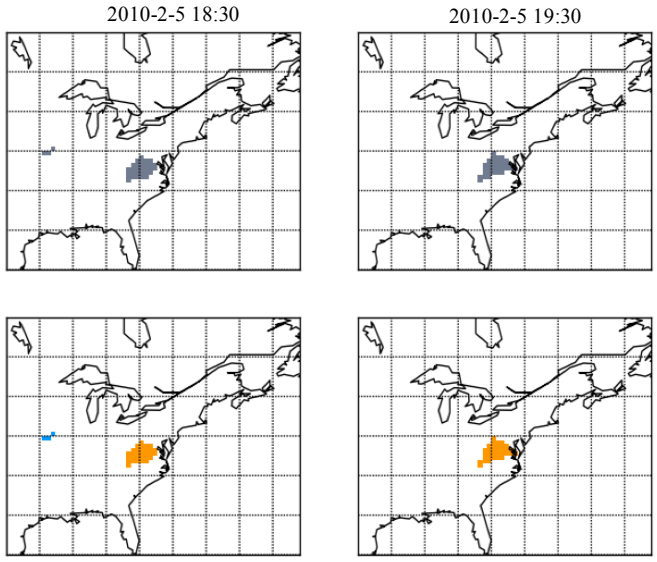

Figure 9. Same events as Fig. 8 but at 1830Z and 1930Z. Note: Only on event persists beyond 1830Z

ACKNOWLEDGMENT

This work was primarily funded by the NASA Earth Science Technology Office (ESTO) through its Advanced Information Systems Technology (AIST) Program. It is also partially supported by the National Science Foundation's (NSF) EarthCube program.

REFERENCES

[1] Paradigm4, "SciDB http://www.paradigm4.com/technology/"

[2] A. Oloso, K.-S. Kuo, T. Clune, P. Brown, and A. Poliakov, "Implementing connected component labeling as a user defined operator for SciDB," 9th Extremely Large Databases Conference (XLDB 2016), 24-26 May 2016, Menlo Park CA, http://www-conf.slac.stanford.edu/xldb2016/.

[3] R. Sedgewick, and K. Wayne, "Algorithms, 4th ed., section 1.5," 2002-2014, http://algs4.cs.princeton.edu/15uf/.

[4] Global Modeling and Assimilation Office (GMAO), "GEOS-5 SYSTEM," https://gmao.gsfc.nasa.gov/GEOS/.

[5] Global Modeling and Assimilation Office (GMAO), "MERRA," https://gmao.gsfc.nasa.gov/GEOS/.

[6] K.-S. Kuo, A. Oloso, M. Bosilovich, A. Collow, M. Rilee, T. Clune, "Thirty-plus (30+) Years of Snowstorm Climatology Obtained from MERRA Ranalysis," 96th AMS Annuak Meeting, 10-14 January 2016, New Orleans, LA, https://annual.ametsoc.org/2016/.

[7] Global Modeling and Assimilation Office (GMAO) (2008), tavg1_2d_flx_Nx: MERRA 2D IAU Diagnostic, Surface Fluxes, Time Average 1-hourly V5.2.0, version 5.2.0, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed June 2015. doi:10.5067/4EQ54AKI405R.

[8] Global Modeling and Assimilation Office (GMAO) (2008), tavg1_2d_lnd_Nx: MERRA 2D IAU Diagnostic, Land Only States and Diagnostics, Time Average 1-hourly V5.2.0, version 5.2.0, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed June 2015. doi:10.5067/YL8Z7MICQZF9.

[9] Global Modeling and Assimilation Office (GMAO) (2008), tavg1_2d_slv_Nx: MERRA 2D IAU Diagnostic, Single Level Meteorology, Time Average 1-hourly V5.2.0, version 5.2.0, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed June 2015. doi: 10.5067/B6DQZQLSFDLH.

[10] Global Modeling and Assimilation Office (GMAO) (2008), const_2d_asm_Nx: MERRA DAS 2d constants V5.2.0, version 5.2.0, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed June 2015. doi:10.5067/HEE4F4IL912I.